DOCUMENT RESUME

ED 393 912 TM 024 913

AUTHOR Singley, Mark K.; Bennett, Randy Elliot
TITLE Toward Computer-Based Performance Assessment in Mathematics.
INSTITUTION Educational Testing Service, Princeton, N.J.
REPORT NO ETS-RR-95-34
PUB DATE Oct 95
NOTE 24p.
PUB TYPE Reports - Evaluative/Feasibility (142)

EDRS PRICE MF01/PC01 Plus Postage.
DESCRIPTORS *Computer Assisted Testing; Item Analysis; *Mathematics Tests; Multiple Choice Tests; *Performance Based Assessment; *Problem Solving; Schemata (Cognition); *Scoring; *Test Construction; Test Items

ABSTRACT

One of the main limitations of the current generation of computer-based tests is its dependency on the multiple-choice item. This research was aimed at extending computer-based testing by bringing limited forms of performance assessment to it in the domain of mathematics. This endeavor involves not only building task types that better reflect valued problem solving; but also creating an integrated set of supports including easy-to-use interfaces; tutorials to teach novice computer users how to negotiate those interfaces; tools that help test developers create items quickly; and mechanisms for scoring constructed responses more efficiently. The cornerstone of the effort's thinking about task generation is the notion of a problem schema, a set of variables and constraints that define a problem's deep structure. A version of a Test Developer's Assistant is being developed that would allow for item creation and tracking, information retrieval, and item analysis in computer-based performance assessment. (Contains two figures, four tables, and seven references.) (Author/SLD)

**********
* Reproductions supplied by EDRS are the best that can be made *
* from the original document. *
**********

RR-95-34

ED 393 912

RESEARCH REPORT

U.S. DEPARTMENT OF EDUCATION
Office of Educational Research and Improvement
EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)

☑ This document has been reproduced as received from the person or organization originating it.

☐ Minor changes have been made to improve reproduction quality.

• Points of view or opinions stated in this document do not necessarily represent official OERI position or policy.

PERMISSION TO REPRODUCE THIS MATERIAL HAS BEEN GRANTED BY

H. I. BRAUN

TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC).

# TOWARD COMPUTER-BASED PERFORMANCE ASSESSMENT IN MATHEMATICS

**Mark K. Singley**
**Randy Elliot Bennett**

**Educational Testing Service**
**Princeton, New Jersey**
**October 1995**

BEST COPY AVAILABLE

TM024913

# Toward Computer-Based Performance Assessment in Mathematics

*Mark K. Singley and Randy Elliot Bennett*

*Educational Testing Service*

*Princeton, NJ 08541*

Copyright © 1995. Educational Testing Service. All rights reserved.

## Abstract

One of the main limitations of the current generation of computer-based tests is its dependency on the multiple-choice item. Our work is aimed at extending computer-based testing by bringing limited forms of performance assessment to it in the domain of mathematics. This endeavor involves not only building task types that better reflect valued problem solving, but also creating an integrated set of supports including easy-to-use interfaces, tutorials to teach novice computer users how to negotiate those interfaces, tools that help test developers create items quickly, and mechanisms for scoring constructed responses more efficiently.

## Toward Computer-Based Performance Assessment in Mathematics

As for most businesses and social services, large-scale testing programs are coming into the technological age. Now available on computer are educational placement tests (the College Board's Computerized Placement Tests and ACT's COMPASS), admissions examinations (the GRE General Test), selection measures (Armed Services Vocational Aptitude Battery) and licensure tests for teachers (Praxis I).

Computer-based tests have important advantages. They offer conveniences to examinees in that they can be taken more frequently than group-administered paper-and-pencil tests, be given in a more comfortable and relaxed setting, and have results unofficially reported immediately upon the conclusion of testing. They offer measurement advantages too: adaptive tests can be shorter than conventional tests without any loss in precision and measure with substantially equal accuracy throughout the score scale.

Although they have clear advantages, computer-based tests have important limitations too. One critical limitation is the dependence of the first generation of such tests on traditional multiple-choice items. As testing programs incorporate new technology, ways must be found to ensure that this technology facilitates rather than limits the assessment of desired skills. Toward this end, we advocate using various delivery and capture technologies to permit use of a wide array of performance tasks (Bennett, 1994). In this view, some performance tasks might be delivered and responded to on computer. In cases where the nature of the task or the quality of the problem solving would be altered by using the computer, we advocate delivering the task in the traditional manner and, if possible, digitally capturing the response so that it can be processed more efficiently (e.g., scanning a handwritten essay).

The current paper describes that aspect of our work aimed at extending computer-based testing so as to bring limited forms of performance assessment to it. In particular, we review the development of performance tasks intended for large-scale testing in mathematics. This work includes not only the performance tasks themselves, but also interfaces for delivering the tasks and capturing the responses, tutorials to teach novice computer users to negotiate this interface, item writing tools for test developers, and mechanisms for scoring item responses.

## Some Simple Performance Tasks

Test questions calling for a numeric entry can be easily presented and scored by computer. Such questions require almost no response processing beyond determining simple mathematical equivalences (e.g., that 1.0 = 1) or possibly applying tolerances for how close to the key value the examinee's response must be to receive credit. Table 1 gives an example from Praxis I, a basic skills test for prospective teachers.

*Table 1*. A numeric entry item from Praxis I.

What is the value of x?

$\frac{24}{72} = \frac{16}{x}$

x = __.

*Note*. Adapted from *Praxis I: Academic Skills Assessments--Tests at a Glance*. Princeton, NJ: Educational Testing Service, 1994.

A logical next step is to pose questions calling for answers that must be rendered in mathematical symbolism, such as expressions or equations. Such items are valuable in that they involve specifying functional relationships between variables and can thereby focus on problem representation as opposed to the more rote aspects of problem solution. Table 2 shows four such problems which, if they could be used on computer-based tests, would not only broaden the diversity of skills that could be assessed in constructed-response form but conceivably tap skills that the traditional multiple-choice item could not as easily accommodate (see item #3 particularly).

Item #1 represents a highly general class of questions for which answers must take the form of numbers or expressions. Item #2 is a special case of #1 in which the answer must be any number or expression drawn from a class of possible answers, where each answer is quantitatively distinct. Item #3 is an extension of #2, asking the examinee to list several such answers from this class. Last, item #4 calls for not only an answer but also the work leading to that answer.

## Interface Designs

The items in Table 2 represent general forms that can be used to test--depending upon the capabilities and intentions of the item writer--a subset of the problem-solving skills that performance assessment was intended to tap or, alternatively, lower-level procedural knowledge. A second critical factor in determining the nature of the skills assessed, however, is how these item types are implemented on computer, in particular how the machine-human interaction is structured.

*Table 2.* Examples of items calling for symbolic answers.

1. A normal line to a curve at a point is a line perpendicular to the tangent line at the point. The equation of the normal line to the curve $y = 2x^2$ at the point $(1,2)$ is given by $y =$ _____.

2. Give an example of a function $f(x)$, for which $f(a+b) = f(a) + f(b)$.

_____.

3. Give up to five examples of a system of two linear equations in x and y that has the solution $x = 3$ and $y = 4$ and for which at least one of the equations has nonzero coefficients for both x and y.

1._____

2._____

3._____

4._____

5._____

4. During a break in its program at 6:15 p.m., a TV station began a fund-raising campaign that has a goal of raising $13,900 through viewer solicitation. It raised the first $7,000 at the rate of $1,200 per minute. When a special appeal began, pledges came in at the rate of $3,000 per minute. If funds continue to come in at the same rate as during the special appeal, until what time, to the nearest minute, must the program break continue in order for the station to reach its goal?

_____

_____

_____

_____

_____

_____

Since computer-based performance assessment is an emerging field, there are no guidelines and few examples on which to base interface designs. One source of information is the intelligent tutoring literature, which describes several interfaces built for collecting mathematical expressions (Brown,1985; McArthur, Stasz, & Hotta, 1986-87; Singley, Anderson, & Gevins, 1991). However, because these interfaces were built to support the requirements of instruction, they offer only limited conceptual guidance for the high-stakes assessment context.

A fundamental tenet of psychological measurement is that, in any test situation, one wants to maximize construct-relevant variance and minimize construct-irrelevant variance. Although the introduction of computer-based tests offers many potential advantages in terms of validity, one potential downside is the introduction of construct-irrelevant variance associated with the use of an unfamiliar interface.

For the purposes of this discussion, we can view test performance as being composed of the operation of two independent subcomponents. First, there is the domain component which is what the test is trying to measure. So, according to our simple model, the examinee first uses domain knowledge and skills to determine the answer to a test question. However, the results generated by this domain component cannot be measured directly through electrodes on the scalp; they must be expressed in some way. So, after determining the answer, the examinee uses expressive skills to share it with the examiner. Of course, for complicated problems involving substeps, the examinee may express parts of the solution as he or she goes along (this raises interesting complications concerning the role of external memory in problem solving). But, even though they are interleaved in time, we can still regard the domain-relevant and expressive components as largely independent.

What are the implications of this simple model of skilled performance for interface design? Minimally, the interface should allow for the expression of all solutions that might be generated by examinees. Otherwise, we have a bad Whorfian interaction of sorts, where the interface is limiting the kinds of solutions that are possible. One might imagine both good and bad solutions that are inexpressible by an interface; both cases seem to be harmful to validity.

Since the expressive component is not the construct of interest, the variance contribution it makes to the total measurement should be minimized. This means ideally all examinees should be equally facile with the interface. The goal of interface design should be to allow all examinees, with minimal training prior to taking the test, achieve more or less asymptotic levels of expressive competence. We essentially want to insure a ceiling effect for the interface component of test performance.

For efficiency of measurement, examinees should spend as small a proportion of their total time expressing their answers as possible, as this allows them to spend relatively more time on construct-relevant activities. To cite an extreme example, an interface which required examinees to draw graphs of equations by filling in individual bits in a bitmap editor might be easy to learn, but it would be unacceptable in terms of the amount of time examinees spent expressing their answers.

In addressing these general concerns, we have been guided by several specific criteria for interface design adapted from Sebrechts, Bennett, and Katz (1993). These criteria are:

1. *Ease of learning*. Instructional applications and productivity tools like word processors and spread sheets are used on a repeated basis, so some amount of time spent

learning to negotiate the software can usually be justified. In contrast to these applications, most individuals take a high-stakes test like the GRE General Test only once. Given the infrequency of use, the time needed for learning must be extremely short.

2. *Ease/Efficiency of use*. In addition to being easy to learn, the interface should be simple to negotiate once it is learned. Entering and editing responses should be straightforward and efficient. Lack of keyboard facility should not impede the examinee.

3. *Problem-solving fidelity*. Because we are primarily interested in assessing rather than engendering problem-solving skill, we wish to observe that proficiency under conditions that are as naturalistic as possible. In paper-and-pencil form, the constraints on problem solving tend to be minimal: Examinees are free to write anywhere on the page in any direction, draw diagrams, formulate syntactically incorrect equations, and do calculations mentally.

4. *Response interpretability*. Whereas loose constraints characterize naturalistic problem solving, they can pose considerable problems for automatic analysis. Thus, some constraints need to be imposed to facilitate machine understanding. The challenge is in finding an appropriate balance that maximally facilitates automatic analysis and does minimal damage to problem-solving fidelity.

5. *Flexibility*. One of the benefits of computer-based delivery is the ability to vary constraints. The interface should permit certain constraints to be switched on and off depending upon the needs of the particular testing program.

6. *Extensibility*. Our interface should be a building block toward a more complete performance assessment system. As such, the interface must be designed and written to be easily extensible.

7. *Consistency with ETS computer-based testing standards*. Among other things, these standards define conventions for machine-user interaction and screen display. Conforming to these standards makes it easier to maintain software across the different testing programs. Examinees also benefit as they must learn only one set of conventions to take an ETS test.

Figure 1 presents the current interface for items whose answers take the form of numbers or single expressions. Many features of the interface promote ease of learning and use. The entry of expressions is completely mouse-driven, so as not to penalize users who are unfamiliar with the keyboard. Digits and arithmetic operators appear in the standard calculator configuration, which makes them easier to find. A menu of alphabetic characters to use as variables is easily accessed by a click of a button. The interface provides for superscript and subscript modes as opposed to having users enter syntactic markers like carats to denote exponentiation. The user simply clicks on the "superscript" button and the next number appears in the superscript position. The interface also displays complex expressions involving division graphically with a horizontal division bar as opposed to the standard slash. The natural, graphical representation of superscripts,

subscripts, and division makes it easier for users to parse expressions they have just entered and minimizes the chances of a mismatch between the system's interpretation of an expression and the user's intention.

The interface imposes certain minimal constraints on the entry of expressions to minimize construct-irrelevant errors and facilitate interpretation and scoring. For example, the interface disables certain buttons on the soft keyboard based on the current entry mode selected. If the user has selected superscript mode, the interface disables the entry of certain mathematical operators like multiplication and division as well as alphabetic characters, as only the digits, +, and - are valid superscript characters. Again, this minimizes the occurrence of what are often simply typing errors and increases the chances we will get an interpretable, scorable response. Also, when users submit their final answer, the interface checks the expression for syntactic correctness. For example, expressions are flagged for the inappropriate juxtaposition of operators (e.g., a multiplication symbol followed immediately by a division symbol) and malformed numbers (e.g., a number containing two decimal points). Also, parentheses must be completely balanced so that the user's intentions are made clear.

The interface is completely generic for symbolic expression entry, and could easily be reused in other testing contexts. This would minimize the learning burden on users taking multiple quantitative tests. Even for the adept computer user, however, such an interface may require orientation. For that reason, a brief tutorial was created to familiarize examinees with the response type. This tutorial introduces the symbol palette and shows how to formulate expressions using features like the subscript and superscript radio buttons and the variable and constants menu. Figure 2 shows one of the screens from this tutorial. For the benefit of novice computer users, more general tutorials are also offered. These tutorials go over using the mouse, scrolling, and other fundamentals needed to take computer-based tests effectively.

## Scoring Mechanisms

An abiding problem that limits the use of performance tasks in large-scale assessments is the high cost of human scoring. We have been working to develop software that can automatically score broad classes of symbolic mathematical responses without human assistance. To build these capabilities, we have drawn on recent developments in symbolic mathematical computation. The subfield of symbolic computation within computer science has made enormous strides in the last two decades. Twenty-five years ago, the simplification and canonicalization of mathematical expressions was not well understood and was considered to be an advanced topic in artificial intelligence. Today, sophisticated algorithms for doing symbolic mathematical transformations have been specified and ongoing work in the field is devoted to the fine-tuning of these algorithms (Norvig, 1993). These algorithms have been packaged in such commercially-available software packages as Mathematica, Maple, and Reduce.

We have been working to adapt these symbolic computation algorithms for assessment purposes. To date, we have developed two capabilities: 1) a system that can dichotomously score single multivariate rational expressions, and 2) a system that can assign full or partial credit scores to extended multi-step responses on problems involving linear systems of equations.

**Scoring Single Expressions**

Many problems are posed in mathematics that involve the construction of a symbolic mathematical expression as the response (e.g. problem 1 in Table 2). We have developed general, accurate, cost-effective, and immediately usable routines that will dichotomously score rational expressions of arbitrary complexity. The routines are general in that they can be used for any test item whose answer is a rational expression. They are accurate in that, once debugged, the algorithms should produce correct results in all cases. They are cost-effective in that entering the key for a new item takes seconds, not the days, weeks, or even months that might be required in a knowledge-based scoring system. Finally, the routines are immediately usable, as evidenced by the fact that they are now being prepared for pilot testing in the new GRE Mathematical Reasoning Test.

As mentioned above, the software can dichotomously score rational expressions, i.e. multivariate polynomials that may or may not involve division. Here are some examples of such expressions:

(1) $x^2+.7x+.12$

(2) $\dfrac{(x^3+y^2)z}{x+y}$

(3) $\dfrac{(m-2p)(n-2p)}{4}$

(4) $-\dfrac{1}{4}x+\dfrac{9}{4}$

A major problem in recognizing the correctness of expressions such as these lies in the fact that there are an infinite number of ways to express the same mathematical relationship. This is the problem of *mathematical paraphrase*. For example, expression (4) above (which happens to be the answer to problem 1 in Table 2) has the following equivalent forms, all of which were entered as correct responses by examinees in a recent field test:

(5) $-\frac{x}{4}+\frac{9}{4}$

(6) $-.25x+2.25$

(7) $\frac{(9-x)}{4}$

(8) $\frac{(-x+9)}{4}$

(9) $2-\frac{1}{4}(x-1)$

As should be apparent, even in this simple case, there are an infinite number of equivalent forms. The program we've constructed uses principles of symbolic computation to reduce equivalent rational expressions to a single normal form. (This normal form, by the way, is far removed from the surface-level representation of these expressions.) Once in this form, the equivalence of expressions can be easily determined.

Another problem plaguing the automatic scoring of rational expressions is the problem of *numerical imprecision*. This problem arises whenever a response (or intermediate forms leading to a response) involves non-integer values and rounding or approximation is required. The routines we have developed allow a test developer to associate an error term (e.g. $10^{-5}$) with each question that specifies the precision of match required in order for a response to be scored as correct.

**Scoring Multiple-Line Responses**

In many assessment situations, it would be useful if it were possible to collect information not only about the final result of an examinee's thinking but also something about the process by which the examinee arrives at that result. Such information could be used for diagnostic purposes or as a basis for assigning partial credit. In mathematics, many problems involve multiple steps which lead to a final solution. When examinees work on paper, many of these intermediate steps are written down, leaving a trace of the solution process which could be subjected to analysis. We have been working to develop algorithms that can analyze such multi-step responses in terms of correctness and completeness.

We have restricted our attention to problems that can be construed as involving linear systems of equations. Many standard algebra word problems fall into this category.

For example, here is an algebra word problem that appeared on a recent SAT in multiple-choice format:

> Excluding rest stops, it took Juanita a total of 10 hours to hike from the base of a mountain to the top and back down again by the same path. If while hiking she averaged 2 kilometers per hour going up and 3 kilometers per hour coming down, how many kilometers was it from the base to the top of the mountain?

The key to our analysis of responses to such problems is the assertion that word problems can be characterized and categorized in terms of the underlying set of equations that relate the entities of the problem to one another. According to this analysis, problems that superficially appear quite distinct may in fact be instances of the same underlying problem structure, or *schema* (Mayer, 1981). The Juanita problem is an instance of the *round-trip* schema. The round-trip schema involves the following equations:

(10) $d_t = d_u + d_d$
(11) $t_t = t_u + t_d$
(12) $d_u = r_u * t_u$
(13) $d_d = r_d * t_d$
(14) $d_t = r_t * t_t$
(15) $d_u = d_d$

This set of variables and primitive equations defines an entire class of problems. Given a set of variables and equations such as this, a particular problem within the class is represented as a set of variable assignments and a goal. Table 3 defines the schema variables and gives their values in the context of the Juanita problem. Table 4 organizes the schema equations into a table. One of the relationships in the table, $r_u + r_d = r_t$, does not hold. However, the other relationships are correct.

*Table 3.* Variables, their meanings, and their values.

| VARIABLE | MEANING | VALUE |
|---|---|---|
| $d_u$ | distance up | $x$ (goal) |
| $d_d$ | distance down | unknown |
| $d_t$ | distance total | unknown |
| $t_u$ | time up | unknown |
| $t_d$ | time down | unknown |
| $t_t$ | time total | 10 h |
| $r_u$ | rate up | 2 km/h |
| $r_d$ | rate down | 3 km/h |
| $r_t$ | rate total | unknown |

*Table 4*. Round-trip schema equation table.

| | part + | part = | whole |
|---|---|---|---|
| distance = | $d_u$ | $d_d$ | $d_t$ |
| rate * | $r_u$ | $r_d$ | $r_t$ |
| time | $t_u$ | $t_d$ | $t_t$ |

In the sample problem, equations (11), (12), (13), and (15) must be instantiated with the given information and composed together to create the equation that provides the solution:

$$(16)\ \frac{d}{2}+\frac{d}{3}=10 \qquad \text{where } d = d_u \text{ or } d_d$$

Using algebra, one can determine from this equation that $d = 12$. Unfortunately from the standpoint of analysis, there are many different ways for an examinee to arrive at such an answer. Furthermore, there are many ways in which an examinee might evidence some level of partial understanding in an errorful solution. For example, an examinee might show the following work when solving the Juanita problem:

(17) $t_u + t_d = t_t$
(18) $t_t = 10$
(19) $2d + 3d = 10$
(20) $d = 2$

This solution is incorrect, but much of the work shown is valid and it therefore may be deserving of partial credit. How do we systematically analyze such responses?

Of course, the problems of mathematical paraphrase and numerical imprecision outlined above still hold in the analysis of multiple-line responses. But in addition to these problems, there are many others:

- Many different strategies for solving a problem exist; all valid strategies must be accommodated in the analysis.
- Substeps may be combined; the examinee is free to determine the granularity of each step.
- Substeps may be skipped; certain calculations or symbolic transformations might be done in the examinee's head.
- Substeps may be inconsistent; one step may directly contradict another.

- Substeps may be partially correct; a single step may be composed of correct and incorrect elements.

- The solution trace may make use of multiple symbol systems (e.g. figures, tables, mathematical symbols, natural language).

For the purposes of the present work, we have assumed that the examinee's response will be composed entirely of syntactically correct expressions or equations involving numbers, mathematical symbols and units, and all the variables that appear in the solution must be defined in terms of their problem referents. We have chosen to defer the analysis of supporting figures and tables for now.

Given these assumptions about the kind of input we receive from examinees, we apply techniques from *constraint logic programming* to analyze the examinee's response. Constraint logic programming is an extension to logic programming (e.g. Prolog) that allows for the declarative representation of a quantitative problem as a set of linear constraints. Once represented in this way, the answer to the problem can be easily *proven* by propagating the constraints. If the problem is underdetermined (i.e. there are an insufficient number of constraints to uniquely determine the solution), the system can derive the most constrained functional relationship between the goal variable and other variables in the problem. We are using constraint logic programming not to solve problems directly but rather to determine the relationship between a canonical representation of the constraints of the problem (the schematic description) and an examinee's response.

As outlined above, we first develop a structural, schematic description of the problem we are analyzing. This description is composed of a set of primitive equations, a set of given variables, and a goal variable. With this problem description in hand, we do a two-pass analysis of the response:

**Correctness:** We first determine whether the response is correct, i.e. whether what is presented in the response is consistent with the problem description. To do this, we try to determine whether the response can be derived from the problem description. We frame the problem in terms of a logical proof: Given the problem description, is it possible to *prove* the response? If it is, we at least know that the equations and values of variables presented in the response are consistent with the problem description. However, we still do not know whether the examinee has represented *all* of the problem constraints in the response and has derived the final answer.

If we determine that the examinee's response is *not* correct, we then systematically vary the constraints in the problem description and try again to prove the correctness of the response. In other words, we try alternative sets of premises (alternative problem descriptions) and see whether any of these sets can prove the conclusion (the response). For example, we may propose a different problem

description that replaces all primitive constraints of the type $d = rt$ (equations 12, 13, and 14) with the errorful variant $t = rd$. If, after degrading the problem description in this way, we are successful in proving the response, we have a very specific account of the error in the response: We know which constraint has been misrepresented and in what way. (It turns out that in the sample response shown above, two errors of this type are present in line 19.)

**Completeness:** Once the analysis for correctness is completed, we must analyze the response for completeness. To determine whether the response is complete, we simply reverse the premises and conclusions of our earlier proof: Given the examinee's response, is it possible to *prove* the problem description? If it is, then we know that at some level all of the relevant constraints are represented in the response, and the response is complete. If it is not, we attempt to determine what is missing in the response. We do this through a procedure very similar to the one outlined above for correctness: We systematically supplement the response with constraints from the problem description and attempt to determine the minimal set of constraints from the problem description that, when added to the response, make it possible to solve the problem. In this way, we can once again pinpoint precisely which constraints are missing from the examinee's response. For example, consider the following correct yet incomplete response to the Juanita problem:

(21) $d_u = 2t_u$
(22) $d_d = 3t_d$
(23) $2t_u = 3t_d$

All of the work shown is correct, but the examinee has failed to provide all of the constraints necessary to uniquely determine the value of $d_u$ or $d_d$. By systematically supplementing this response with constraints drawn from the problem description, we can determine that, with the addition of the constraints $t_t = t_u + t_d$ (equation 11 above) and $t_t = 10$, the problem is solvable. Thus, once again we have a precise description of what is present and what is missing in the response.

In sum, our approach involves representing a particular problem as a set of constraints, and then determining which (if any) of those constraints is violated and/or absent in the response. By reducing a particular response to the set of quantitative constraints it represents, the analysis bypasses many of the problems outlined earlier concerning variations in the surface forms of responses: multiple strategies, skipped steps, composed steps, and partially correct steps. According to this approach, two responses are equivalent if they can be reduced to the same set of primitive constraints.

Such an approach lends itself well to the implementation of partial credit scoring rubrics. Subscores can be associated with each constraint in the problem description. The presence or absence of the constraint in the response can either increment a base score or

decrement a total score for the problem, respectively. Also, subscores can be associated with common errors, i.e. common degradations of the problem description. In addition, such rubrics need not be created for each individual problem; they can be created once and for all for an entire class of problems if they are done at the level of the problem schema.

Currently, we are putting the finishing touches on a prototype system that performs the kinds of analyses described here. We are about to subject the system to a formative evaluation that will involve an in-depth analysis of hundreds of multi-line responses collected in a new experimental version of the SAT II Mathematics Achievement Test. This test was administered on paper, so we will be transcribing the examinees' responses into electronic format. (This involves making minimal assumptions about the kind of interface the examinees might use.) In our evaluation, we will be comparing the partial-credit scores assigned by our system with those assigned by human judges.

## Tools for Item Development

We are just beginning to apply some of the ideas we have been pursuing in the realm of automatic scoring to the task of item generation. Once again, the cornerstone of our thinking is the notion of a problem schema, a set of variables and constraints that define a problem's deep structure. Given this notion of a schema, a wide spectrum of item generation possibilities exists which range from the deep to the superficial:

- Given a set of primitives (e.g. $d = rt$, $d_1 + d_2 = d_t$, etc.), generate a set of interesting problem schemas. These schemas could be arranged into a taxonomy of sorts, and could serve as a repository of ideas for test developers.

- Given a particular schema (e.g. the round-trip schema defined above), generate interesting structural elaborations. This involves adding one or more constraints. For example, in the round-trip schema, one could add some relationship between the rate for the first leg and the rate for the second.

- Given a particular schema, generate all the possible problem structures. A problem structure is defined as a particular configuration of given and goal variables.

- Given a problem structure, generate and/or select a problem context. Thus, the round-trip schema could be instantiated in a hiking context, a boating context, etc.

- Given a context, vary the noun referents for problem variables and generate different sets of values for given variables. For example, Juanita becomes Juan and the total time changes from 10 to 20 hours.

- Given a context, noun referents and variable values, generate a natural language cover story.

We envision embodying the above functionality in an integrated system that offers more to the test developer than just item generation capabilities. We have been developing a vision of a Test Developer's Assistant that, in addition to semi-automatic item generation, would offer three other distinct types of functionality:

1. *Item creation and tracking.* These are basic word processing functions that form the backbone of functionality for the system. Users should be able to type in items, edit them, import graphics, etc. Items that are created with the system should be compatible with test delivery software. Item management features should allow the test developer to track the location and status of an item throughout its life cycle.

2. *Information retrieval.* While authoring new items, test developers should be able to search both local and remote sources for ideas and raw materials for items (text, figures, etc.). Part of this effort may actually entail putting useful and frequently used sources online ourselves (e.g. math textbooks). The material retrieved through searches should be easily and directly usable by the item creation tools, e.g. through cutting and pasting. A special case of the above effort would be gathering together existing instances of items and putting them online in a searchable or browsable form. When available, item statistics should be stored with each item and should be a searchable feature (e.g. a possible search is to retrieve all instances of word problems within a certain range of difficulty). In addition, items would be indexed in terms of whatever conceptual frameworks are used in either the automatic generation or analysis of items. Again, the text/graphics of library items should be directly accessible by the item creation tools, i.e. the user should be able to edit the text of a library item to create a new item.

3. *Item analysis.* Once an item is reasonably complete (regardless of how much machine assistance is rendered in its creation), it can be subjected to an analysis that will attempt to predict its level of difficulty. This analysis will be based on the availability of psychological theory that relates features of the item to item statistics. Depending upon the set of features for a particular item type, the analysis may proceed completely automatically or may require some human assistance. One possibility is to use templates while authoring items so that analysis can proceed automatically.

## Conclusion

This paper has described our efforts to bring limited forms of performance assessment to large-scale computer-based testing programs. As should be evident, these efforts are substantial, involving not only development of the open-ended tasks but also creation of various supporting tools. It is worth reiterating that, in contrast to the performance tasks used in some paper-and-pencil testing programs, ours are very limited: the test items are relatively discrete and the responses have comparatively few

components. Delivering more complex computer-based tasks will undoubtedly require greater levels of effort. Just as clearly, the effort required to deliver performance tasks in computer-based testing programs will at some point exceed the benefits that can be derived. Whereas our work has given us a much better understanding of the costs involved, judgments of the benefits will take longer. The case for computer-based performance assessment in mathematics, as elsewhere, will need to rest on evidence that these tasks measure valued mathematical problem-solving skills effectively, are fair to population subgroups, are favorably viewed by examinees and decision makers, and have positive (or at least no negative) educational effects.

## References

Bennett, R. E. (1994). An electronic infrastructure for a future generation of tests (RR-94-61). Princeton, NJ: Educational Testing Service.

Brown, J. S. (1985). Process versus product: A perspective on tools for communal and informal electronic learning. Journal of Educational Computing Research, 1, 179-201.

McArthur, D., Stasz, C., & Hotta, J. Y. (1986-87). Learning problem-solving skills in algebra. Journal of Educational Technology Systems, 18, 303-324.

Mayer, R. E. (1981). Frequency norms and structural analysis of algebra story problems into families, categories, and templates. Instructional Science, 10, 135-175.

Norvig, P. (1993). Paradigms of artificial intelligence programming. San Mateo, CA: Morgan Kaufmann.

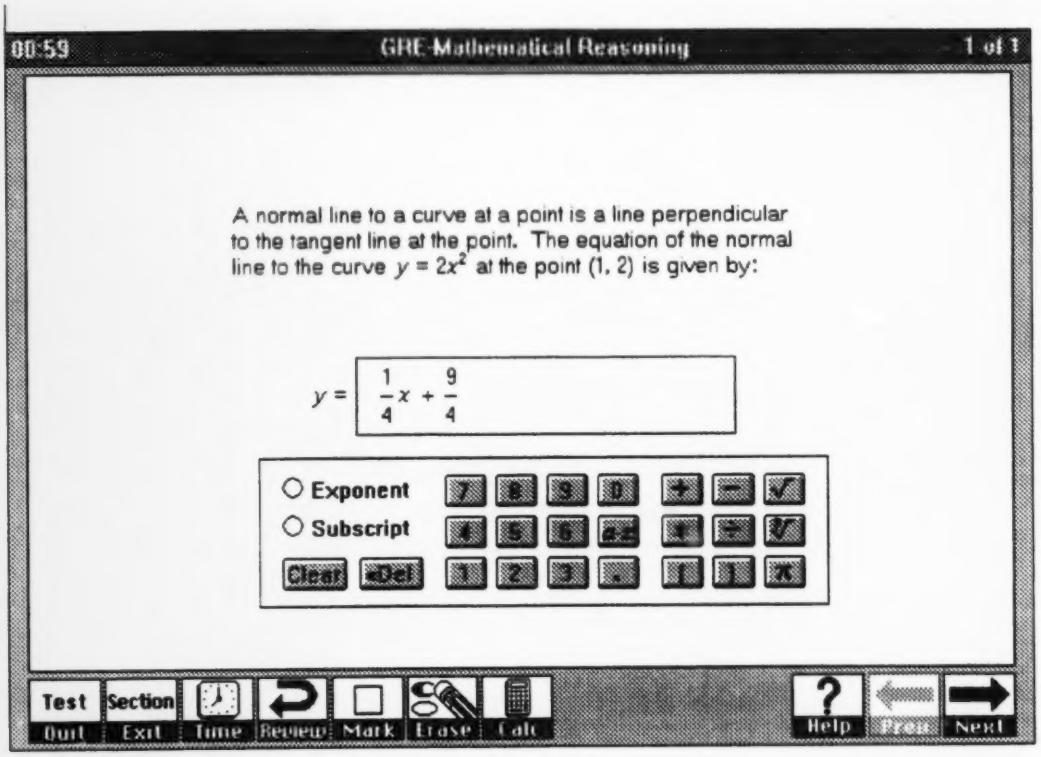
Sebrechts, M. M., Bennett, R. E., & Katz, I. R. (1993). A research platform for interactive performance assessment in graduate education (RR-93-08). Princeton, NJ: Educational Testing Service.

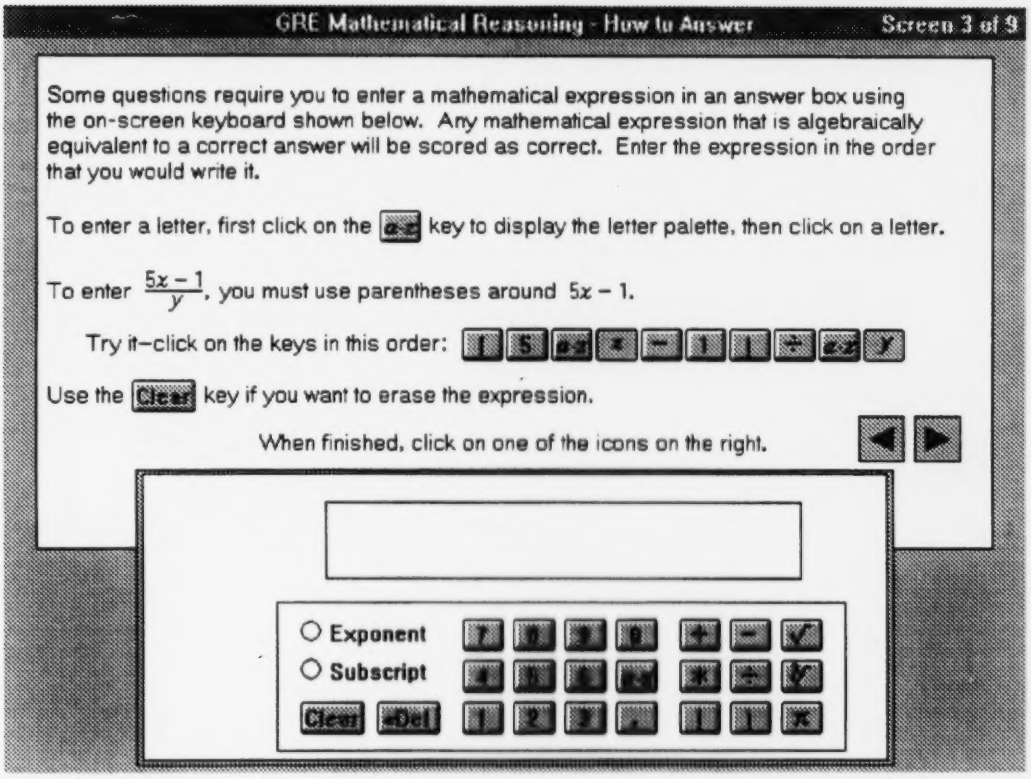
Singley, J. K., Anderson, J. R., & Gevins, J. S. (1991). Promoting abstract strategies in algebra word problem solving. In Proceedings of the International Conference on the Learning Sciences. Charlottesville, VA: Association for the Advancement of Computing in Education.

# Figure Captions

Figure 1. An interface for presenting mathematical problems requiring numeric entries or single expressions.

Figure 2. A tutorial screen for teaching examinees how to enter responses that take the form of mathematical expressions.


00:59
GRE Mathematical Reasoning
1 of 1
A normal line to a curve at a point is a line perpendicular to the tangent line at the point. The equation of the normal line to the curve $y = 2x^2$ at the point (1, 2) is given by:
$y = \frac{1}{4}x + \frac{9}{4}$
Exponent
Subscript
Clear
Del
Test
Quit
Section
Exit
Time
Review
Mark
Erase
Calc
Help
Prev
Next


GRE Mathematical Reasoning - How to Answer
Screen 3 of 9
Some questions require you to enter a mathematical expression in an answer box using the on-screen keyboard shown below. Any mathematical expression that is algebraically equivalent to a correct answer will be scored as correct. Enter the expression in the order that you would write it.
To enter a letter, first click on the key to display the letter palette, then click on a letter.
To enter (5x − 1)/y, you must use parentheses around 5x − 1.
Try it–click on the keys in this order:
Use the Clear key if you want to erase the expression.
When finished, click on one of the icons on the right.
Exponent
Subscript
Clear
Del



BEST COPY AVAILABLE